MIL-STD 1782

10 MAY 1984

MILITARY STANDARD

TELNET PROTOCOL

NO DELIVERABLE DATA

REQUIRED BY THIS DOCUMENT

IPSC/SLHC/TCTS

DEPARTMENT OF DEFENSE
WASHINGTON, D.C. 20301

TELNET Protocol

MIL-STD-1782

1. This Military Standard is approved for use by all Departments and Agencies of the Department of Defense.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to: Defense Communications Agency, ATTN: J110, 1860 Wiehle Avenue, Reston, Virginia 22090, by using the self-addressed Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document, or by letter.

3. Because of the rapid development in this standardization area, an alternative method of communication is offered. Forward responses using the MILNET to DCA-IAS @ DCA-EMS. Cooperation of the user is important to make this protocol meet Department of Defense needs.

FOREWORD

This document specifies the TELNET protocol and a number of approved options. It provides a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication (“linking”) and process-process communication (distributed computation).

CONTENTS

CONTENTS - Continued

CONTENTS - Continued

FIGURES

1. SCOPE

1.1 Purpose. This standard establishes criteria for the TELNET Protocol which supports the standard method of interfacing terminal devices and terminal-oriented processes to each other.

1.2 Organization. This standard introduces the TELNET Protocols role, defines the services provided to users, and specifies the mechanisms needed to support those services. This standard also includes an appendix of options which can be implemented in the TELNET Protocol and a glossary of terms and abbreviations.

1.3 Application. This TELNET Protocol is approved for use in all DoD packet switching networks which connect or have the potential for utilizing connectivity across network and subnetwork boundaries and which require a virtual terminal service. The term network as used herein includes Local Area Networks.

2. REFERENCED DOCUMENTS

2.1 Issues of documents. The following documents of the issue in effect on date of invitation for bids or request for proposal, form a part of this standard to the extent specified herein.

STANDARDS
FEDERAL

FED-STD-1037	Glossary of Telecommunications Terms

MILITARY

MIL-STD-1778	Transmission Control Protocol

2.2 Other Publications. The following documents form a part of this standard to the extent specified herein. Unless otherwise indicated, the issue in effect on date of invitation for bids or request for proposal shall apply. (The provisions of this paragraph are under consideration.)

3. DEFINITIONS

3.1 Definition of terms. The definition of terms used in this standard shall comply with FED-STD-1037. Terms and definitions unique to MIL-STD-1782 are contained herein. (The provisions of this paragraph are under consideration.)

3.2 Definitions of acronyms used in this standard. The following acronyms used in this Military Standard are defined as follows:

a. AO - Abort Output Command
b. AYT - Are You There Command
c. BS - Back Space
d. CR - Carriage Return
e. DDN - Defense Data Network
f. DM - Data Mark
g. DoD - Department of Defense
h. DODIIS - DoD Intelligence Information System
I. EC - Erase Character Command
j. EL - Erase Line Command
k. FF - Form Feed
1. GA - Go-Ahead Command
m. HT - Horizontal Tab
n. IAC - Interpret As Command
o. IBM - International Business Machines, Inc.
P. IP - Interrupt Process Command
q. LF - Line Feed
r. NVT - Network Virtual Terminal
s. TCP - Transmission Control Protocol
t. TELNET - Telecommunications Network
u. VT - Vertical Tab
v. ASCII - American Standard Code for Information Interchange

4. GENERAL REQUIREMENTS

4.1 Introduction. The purpose of the TELNET Protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation). The Appendices to this volume contain DoD approved and supported TELNET options. It is recommended, but not mandatory, that these options be implemented as useful adjuncts to the TELNET protocol. If implemented, they are required to be implemented as published herein.

4.2 General considerations. A TELNET connection is a Transmission Control Protocol (TCP) connection used to transmit data with interspersed TELNET control information. The TELNET Protocol is built upon three main ideas: first, the concept of a Network Virtual Terminal; second, the principle of negotiated options; and third, a symmetric view of terminals and processes.

4.2.1 Network Virtual Terminal (NVT). When a TELNET connection is first, established, each end is assumed to originate and terminate at a Network Virtual Terminal (NVT). An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal. This eliminates the need for "server" and "user" hosts to keep information about the characteristics of each other's terminals and terminal handling conventions. All hosts, both user and server, map their local device characteristics and conventions so as to appear to be dealing with an NVT over the network, and each can assume a similar mapping by the other party. The NVT is intended to strike a balance between being overly restricted (not providing hosts a rich enough vocabulary for mapping into their local character sets), and being overly inclusive (penalizing users with modest terminals). The "user" host is the host to which the physical terminal is normally attached, and the "server" host is the host which is normally providing some service. As an alternate point of view, applicable even in terminal-to-terminal or process-to-process communications, the "user" host is the host which initiated the communication.

4.2.2 Principle of negotiated options. The principle of negotiated options takes cognizance or the tact that many hosts will wish to provide additional services over and above those available within an NVT, and many users will have sophisticated terminals and would like to have elegant, rather than minimal, services. Independent of, but structured within the TELNET Protocol are various "options" that will be sanctioned and may be used with the “DO, DON’T, WILL, WON’T” structure (discussed below) to allow a user and server to agree to use a more elaborate (or perhaps just different) set of conventions for their TELNET connection. Such options could include changing the character set, the echo mode, etc. The basic strategy for setting up the use of options is to have either party (or both) initiate a request that some option take effect. The other party may then either accept or reject the request. If the request is accepted the option immediately takes effect; if it is rejected the associated aspect of the connection remains as specified for an NVT. Clearly, a party may always refuse a request to enable, and must never refuse

a request to disable some option since all parties must be prepared to support the NVT. The syntax of option negotiation has been set up so that if both parties request an option simultaneously, each will see the other's request as the positive acknowledgment of its own.

4.2.3 Symmetry of the negotiation syntax. The symmetry of the negotiation syntax can potentially lead to nonterminating acknowledgment loops -- each party seeing the incoming commands not as acknowledgments but as new requests which must be acknowledged. To prevent such loops, the following rules prevail:

a. Parties may only request a change in option status; i.e., a party may not send out a "request" merely to announce what mode it is in.

b. If a party receives what appears to be a request to enter some mode it is already in, the request should not be acknowledged. This non-response is essential to prevent endless loops in the negotiation. It is required that a response be sent to requests for a change of mode -- even if the mode is not changed.

c. Whenever one party sends an option command to a second party, whether as a request or an acknowledgment, and use of the option will have any effect on the processing of the data being sent from the first party to the second, then the command must be inserted in the data stream at the point where it is desired that it take effect. Some time will elapse between the transmission of a request and the receipt of an acknowledgment, which may be negative. Thus, a host may wish to buffer data, after requesting an option, until it learns whether the request is accepted or rejected, in order to hide the “uncertainty period” from the user.

4.2.4 Use of options. Option requests are likely to flurry back and forth when a TELNET connection is first established, as each party attempts to get the best possible service from the other party. Beyond that, however, options can be used to dynamically modify the characteristics of the connection to suit changing local conditions. For example, the NVT, as will be explained later, uses a transmission discipline well suited to the many “line at a time” applications such as BASIC, but poorly suited to the many “character at a time” applications such as NLS. A server might elect to devote the extra processor overhead required for a “character at a time” discipline when it was suitable for the local process and would negotiate an appropriate option. However, rather than then being permanently burdened with the extra processing overhead, it could switch (i.e., negotiate) back to NVT when the detailed control was no longer necessary. It is possible for requests initiated by processes to stimulate a nonterminating request loop if the process responds to a rejection by merely re-requesting the option. To prevent such loops from occurring, rejected requests should not be repeated until something changes. Operationally, this can mean the process is running a different program, or the user has given another command, or whatever makes sense in the context of the given process and the given option. A

good rule of thumb is that a re-request should only occur as a result of subsequent information from the other end of the connection or when demanded by local human intervention. Option designers should not feel constrained by the somewhat limited syntax available for option negotiation. The intent of the simple syntax is to make it easy to have options -- since it is correspondingly easy to profess ignorance about them. If some particular option requires a richer negotiation structure than possible within “DO, DON’T, WILL, WON’T”, the proper tack is to use “DO, DON’T, WILL, WON’T” to establish that both parties understand the option, and once this is accomplished a more exotic syntax can be used freely. For example, a party might send a request to alter (establish) line length. If it is accepted, then a different syntax can be used for actually negotiating the line length -- such a “sub-negotiation” might include fields for minimum allowable, maximum allowable and desired line lengths. The important concept is that such expanded negotiations should never begin until some prior (standard) negotiation has established that both parties are capable of parsing the expanded syntax.

4.2.5 Synopsis. In summary, WILL XXX is sent, by either party, to indicate that party's desire (offer) to begin performing option XXX, DO XXX and DON'T XXX being its positive and negative acknowledgments; similarly, DO XXX is sent to indicate a desire (request) that the other party (i.e., the recipient of the DO) begin performing option XXX, WILL XXX and WON'T XXX being the positive and negative acknowledgments. Since the NVT is what is left when no options are enabled, the DON'T and WON'T responses are guaranteed to leave the connection in a state which both ends can handle. Thus, all hosts may implement their TELNET processes to be totally unaware of options that are not supported, simply returning a rejection to (i.e., refusing) any option request that cannot be understood. As much as possible, the TELNET protocol has been made server-user symmetrical so that it easily and naturally covers the user-user (linking) and server-server (cooperating processes) cases. It is hoped, but not absolutely required, that options will further this intent. In any case, it is explicitly acknowledged that symmetry is an operating principle rather than an ironclad rule. Standard TELNET options referenced in this document are contained in Appendices A-F.

4.3 The Network Virtual Terminal. The Network Virtual Terminal (NVT) is a bi-directional character device. The NVT has a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which is sent over the TELNET connection and, if “echoes” are desired, to the NVT's printer as well. “Echoes” will not be expected to traverse the network (although options exist to enable a “remote” echoing mode of operation, no host is required to implement this option). The code set is seven-bit USASCII in an eight-bit field, except as modified herein. Any code conversion and timing considerations are local problems and do not affect the NVT.

4.3.1 Transmission of data. Although a TELNET connection through the network is intrinsically full duplex, the NVT is to be viewed as a half-duplex device operating in a line-buffered mode. That is, unless and until options are negotiated to the contrary, the following default conditions pertain to the transmission of data over the TELNET connection:

4.3.1.1 Accumulation of data. Insofar as the availability of local buffer space permits, data should be accumulated in the host where it is generated until a complete line of data is ready for transmission, or until some locally-defined explicit signal to transmit occurs. This signal could be generated either by a process or by a human user. The motivation for this rule is the high cost, to some hosts, of processing network input interrupts, coupled with the default NVT specification that “echoes” do not traverse the network. Thus, it is reasonable to buffer some amount of data at its source. Many systems take some processing action at the end of each input line (even line printers or card punches frequently tend to work this way), so the transmission should be triggered at the end of a line. On the other hand, a user or process may sometimes find it necessary or desirable to provide data which does not terminate at the end of a line; therefore implementers are cautioned to provide methods of locally signaling that all buffered data should be transmitted immediately.

4.3.1.2 TELNET Go Ahead (GA) command. When a process has completed sending data to an NVT printer and has no queued input from the NVT keyboard for further processing (i.e., when a process at one end of a TELNET connection cannot proceed without input from the other end), the process must transmit the TELNET Go Ahead (GA) command. This rule is not intended to require that the TELNET GA command be sent from a terminal at the end of each line, since server hosts do not normally require a special signal (in addition to end-of-line or other locally-defined characters) in order to commence processing. Rather, the TELNET GA is designed to help a user's local host operate a physically half duplex terminal which has a "lockable" keyboard such as the IBM 2741. A description of this type of terminal may help to explain the proper use of the GA command. The terminal-computer connection is always under control of either the user or the computer. Neither can unilaterally seize control from the other; rather the controlling end must relinquish its control explicitly. At the terminal end, the hardware is constructed so as to relinquish control each time that a "line" is terminated (i.e., when the “New Line” key is typed by the user). When this occurs, the attached (local) computer processes the input data, decides if output should be generated, and if not returns control to the terminal. If output should be generated, control is retained by the computer until all output has been transmitted. The difficulties of using this type of terminal through the network should be obvious. The "local" computer is no longer able to decide whether to retain control after seeing an end-of-line signal or not; this decision can only be made by the "remote" computer which is processing the data. Therefore, the TELNET GA command provides a mechanism whereby the "remote" (server) computer can signal the "local" (user) computer that it is time to pass control to the user of the terminal. It should be transmitted at those times, and only at those times, when the user should be given control of the terminal. Note that premature transmission of the GA command may result in the blocking of output, since the user is likely to assume that the transmitting system has paused, and therefore he will fail to turn the line around manually.

4.3.2 Transmission caution. The foregoing, of course, does not apply to the user-to-server direction of communication. In this direction, GAs may be sent at any time, but need not ever be sent. Also, if the TELNET connection is being used for process-to-process communication, GAs need not be sent in either direction. Finally, for terminal-to-terminal communication, GAs may be required in neither, one, or both directions. If a host plans to support terminal-to-terminal communication it is suggested that the host provide the user with a means of manually signaling that it is time for a GA to be sent over the TELNET connection; this, however, is not a requirement on the implementer of a TELNET process. The symmetry of the TELNET model requires an NVT at each end of the TELNET connection, at least conceptually.

4.4 Standard representation of control functions. As stated in paragraph 4.1, the primary goal of the TELNET protocol is the provision of a standard interfacing of terminal devices and terminal-oriented processes through the network. Early experiences with this type of interconnection have shown that certain functions are implemented by most servers, but that the methods of invoking these functions differ widely. For a human user who interacts with several server systems, these differences are highly frustrating. TELNET, therefore, defines a standard representation for five of these functions, as described below. Theme standard representations have standard, but not required, meanings (with the exception that the Interrupt Process (IP) function may be required by other protocols which use TELNET); that is, a system which does not provide the function to local users need not provide it to network users and may treat the standard representation for the function as a No-operation. On the other hand, a system which does provide the function to a local user is obliged to provide the same function to a network user who transmits the standard representation for the function.

4.4.1 Interrupt process (IP). Many systems provide a function which suspends, interrupts, aborts, or terminates the operation of a user process. This function is frequently used when a user believes his process is in an unending loop, or when an unwanted process has been inadvertently activated. IP is the standard representation for invoking this function. It should be noted by implementers that IP may be required by other protocols which use TELNET, and-therefore should be implemented if these other protocols are to be supported.

4.4.2 Abort output (AO). Many systems provide a function which allows a process, which is generating output, to run to completion (or to reach the same stopping point it would reach if running to completion) but without sending the output to the user's terminal. Further, this function typically clears any output already produced but not yet actually printed (or displayed) on the user's terminal. AO is the standard representation for invoking this function. For example, some subsystem might normally accept a user’s command, send a long text string to the user's terminal in response, and finally signal readiness to accept the next command by sending a “prompt” character (preceded by <CR><LF>) to the user's terminal. If the AO were received during the transmission of the text string, a reasonable implementation would be to suppress the remainder of the text string, but transmit the prompt character and the preceding <CR><LF>. (This is possibly in distinction to the action which might be taken if an IP were received; the IP might

cause suppression of the text string and an exit from the subsystem.) It should be noted, by server systems which provide this function, that there may be buffers external to the system (in the network and the user's local host) which should be cleared; the appropriate way to do this is to transmit the “Synch” signal (described below) to the user system.

4.4.3 Are you there (AYT). Many systems provide a function which provides the user with some visible (e.g., printable) evidence that the system is still up and running. This function may be invoked by the user when the system is unexpectedly “silent” for a long time, because of the unanticipated (by the user) length of a computation, an unusually heavy system load, etc. AYT is the standard representation for invoking this function.

4.4.4 Erase character (EC). Many systems provide a function which deletes the last preceding undeleted character or “print position” from the stream of data being supplied by the user. A “print position” may contain several characters which are a result of overstrikes, or of sequences such as <char1> BS <char2>... This function is typically used to edit keyboard input when typing mistakes are made. EC is the standard representation for invoking this function.

4.4.5 Erase line (EL). Many systems provide a function which deletes all the data in the current “line” of input. This function is typically used to edit keyboard input. EL is the standard representation for invoking this function.

4.5 The TELNET “Synch” signal. Most time-sharing systems provide mechanisms which allow a terminal user to regain control of a “runaway” process; the IP and AO functions described above are examples of these mechanisms. Such systems, when used locally, have access to all of the signals supplied by the user, whether these are normal characters or special “out of band” signals such as those supplied by the teletype "BREAK' key or the IBM 2741 “ATTN” key. This is not necessarily true when terminals are connected to the system through the network; the network's flow control mechanisms may cause such a signal to be buffered elsewhere, for example in the user's host. To counter this problem, the TELNET “Synch” mechanism is introduced. A Synch signal consists of a TCP Urgent notification, coupled with the TELNET command DATA MARK. The Urgent notification, which is not subject to the flow control pertaining to the TELNET connection, is used to invoke special handling of the data stream by the process which receives it. In this mode, the data stream is immediately scanned for “interesting” signals as defined below, discarding intervening data. The TELNET command DATA MARK (DM) is the synchronizing mark in the data stream which indicates that any special signal has already occurred and the recipient can return to normal processing of the data stream. The Synch is sent via the TCP send operation with the Urgent flag set and the DM as the last (or only) data octet.

4.5.1 Sending several Synchs. When several Synchs are sent in rapid succession, the Urgent notifications may be merged. It is not possible to count Urgents since the number received will be less than or equal the number sent. When in normal mode, a DM is a no operation; when in urgent mode, it signals the end of the urgent processing. If TCP indicates the end of Urgent data before the DM is found, TELNET should continue the special handling of the data stream until the DM is found. If TCP indicates more Urgent data after the DM is found, it can only be because of a subsequent Synch. TELNET should continue the special handling of the data strewn until another DM is found.

4.5.2 Interesting signals. “Interesting" signals are defined to be: the TELNET standard representations of IP, AO, and AYT (but not EC or EL); the local analogs of these standard representations (if any); all other TELNET commands; other site-defined signals which can be acted on without delaying the scan of the data stream.

4.5.3 One effect of the Synch mechanism. Since one effect of the Synch mechanism is the discarding of essentially all characters (except TELNET commands) between the sender of the Synch and its recipient, this mechanism is specified as the standard way to clear the data path when that is desired. For example, if a user at a terminal causes an AO to be transmitted, the server which receives the AO (if it provides that function at all) should return a Synch to the user.

4.5.4 TELNET command requirement. Just as the TCP Urgent notification is needed at the TELNET level as an out-of-band signal, so other protocols which make use of TELNET may require a TELNET command which can be viewed as an out-of-band signal at a different level. By convention the sequence [IP, Synch] is to be used as such a signal. For example, suppose that some other protocol, which uses TELNET, defines the character string STOP analogously to the TELNET command AO. Imagine that a user of this protocol wishes a server to process the STOP string, but the connection is blocked because the server is processing other commands. The user should instruct his system to:

a. Send the TELNET IP character;

b. Send the TELNET Synch sequence, that is: Send the Data Mark (DM) as the only character in a TCP urgent mode send operation;

c. Send the character string STOP; and

d. Send the other protocols analog of the TELNET DM, if any.

The user (or process acting on his behalf) must transmit the TELNET SYNCH sequence of step b above to ensure that the TELNET IP gets through to the server's TELNET interpreter. The Urgent should wake up the TELNET process; the IP should wake up the next higher level process.

4.6 The NVT printer and keyboard.

4.6.1 NVT printer codes. The NVT printer has an unspecified carriage width and page length and can produce representations of all 95 ASCII graphics (codes 32 through 126). Of the 33 ASCII control codes (O through 31 and 127), and the 128 uncovered codes (128 through 255), the following have specified meaning to the NVT printer:

	NAME	CODE	MEANING
a.	NULL (NUL)	0	No Operation
b.	Line Feed (LF)	10	Moves the printer to the next print line, keeping the same horizontal position.
c.	Carriage Return (CR)	13	Moves the printer to the left margin of the current line.

In addition, the following codes shall have defined, but not required, effects on the NVT printer. Neither end of a TELNET connection may assume that the other party will take, or will have taken, any particular action upon receipt or transmission of the following:

	NAME	CODE	MEANING
d.	BELL (BEL)	7	Produces an audible or visible signal (which does NOT move the print head).
e.	Back Space (BS)	8	Moves the print head one character position towards the left margin.
f.	Horizontal Tab (HT)	9	Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
g.	Vertical Tab (VT)	11	Moves the printer to the next vertical tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
h.	Form Feed (FF)	12	Moves the printer to the top of the next page, keeping the same horizontal position.

All remaining codes do not cause the NVT printer to take any action.

4.6.1.1 Example of code use. The sequence “CR LF”, as defined, will cause the NVT to be positioned at the left margin of the next print line (as would, for example, the sequence “LF CR”). However, many systems and terminals do not treat CR and LF independently, and will have to go to some effort to simulate their effect. (For example, some terminals do not have a CR independent of the LF, but on such terminals it may be possible to simulate a CR by backspacing.) Therefore, the sequence “CR LF” must be treated as a single “new line” character and used whenever their combined action is intended; the sequence “CR NUL" must be used where a carriage return alone is actually desired; and the CR character must be avoided in other contexts. This rule gives assurance to systems which must decide whether to perform a “new line” function or a multiple-backspace that the TELNET stream contains a character following a CR that will allow a rational decision. Note that “CR LF” or “CR NUL” is required in both directions (in the default ASCII mode), to preserve the symmetry of the NVT model. Even though it may be known in some situations (e.g., with remote echo and suppress go ahead options in effect) that characters are not being sent to an actual printer, nonetheless, for the sake of consistency, the protocol requires that a NUL be inserted following a CR not followed by a LF in the data stream. The converse of this is that a NUL received in the data stream after a CR (in the absence of options negotiations which explicitly specify otherwise) should be stripped out prior to applying the NVT to local character set mapping.

4.6.1.2 ASCII code generation. The NVT keyboard has keys, or key combinations, or key sequences, for generating all 128 ASCII codes. Note that although many have no effect on the NVT printer, the NVT keyboard is capable of generating them.

4.6.1.3 Additional codes. In addition to these codes, the NVT keyboard shall be capable of generating the following additional codes which, except as noted, have defined, but not required, meanings. The actual code assignments for these “characters” are in the TELNET Command section, because they are viewed as being, in some sense, generic and should be available even when the data stream is interpreted as being some other character set.

4.6.1.3.1 Synch. This key allows the user to clear his data path to the other party. The activation of this key causes a DM (see command section) to be sent in the data stream and a TCP Urgent notification is associated with it. The pair DM-Urgent is to have required meaning as defined previously.

4.6.1.3.2 Break (BRK). This code is provided because it is a signal outside the ASCII set which is currently given local meaning within many systems. It is intended to indicate that the Break Key or the Attention Key was hit. Note, however, that this is intended to provide a 129th code for systems which require it, not as a synonym for the IP standard representation.

4.6.1.3.3 Interrupt process (IP). Suspend, interrupt, abort or terminate the process to which the NVT is connected. Also, part of the out-of-band signal for other protocols which use TELNET.

4.6.1.3.4 Abort output (AO). Allow the current process to (appear to) run to completion, but do not send its output to the user. Also, send a Synch to the user.

4.6.1.3.5 Are you there (AYT). Send back to the NVT some visible (i.e., printable) evidence that the AYT was received.

4.6.1.3.6 Erase character (EC). The recipient should delete the last preceding undeleted character or "print position" from the data stream.

4.6.1.3.7 Erase line (EL). The recipient should delete characters from the data stream back to, but not including, the last "CR LF" sequence sent over the TELNET connection.

4.6.1.3.8 Intent of additional codes. The spirit of these "extra" keys, and also the printer format effectors, is that they should represent a natural extension of the mapping that already must be done from "NVT" into "local". Just as the NVT data byte 68 (104 octal) should be mapped into whatever the local code for "uppercase D" is, so the EC character should be mapped into whatever the local "Erase Character" function is. Further, just as the mapping for 124 (174 octal) is somewhat arbitrary in an environment that has no "vertical bar" character, the EL character may have a somewhat arbitrary mapping (or none at all) if there is no local "Erase Line" facility. Similarly for format effectors: if the terminal actually does have a "Vertical Tab", then the mapping for VT is obvious, and only when the terminal does not have a vertical tab should the effect of VT be unpredictable.

4.7 TELNET command structure. All TELNET commands consist of at least a two byte sequence: the "Interpret as Command" (IAC) escape character followed by the code for the command. The commands dealing with option negotiation are three byte sequences, the third byte being the code for the option referenced. This format was chosen so that as more comprehensive use of the "data space" is made -- by negotiations from the basic NVT, of course -- collisions of data bytes with reserved command values will be minimized, all such collisions requiring the inconvenience, and inefficiency, of "escaping" the data bytes into the stream. With the current set-up, only the IAC need be doubled to be sent as data, and the other 255 codes may be passed transparently.

4.7.1 TELNET commands defined. The following are the defined TELNET commands. Note that these codes and code sequences have the indicated meaning only when immediately preceded by an IAC.

		CODE	
a.	SE	240	End of subnegotiation parameters.
b.	NOP	241	No operation.
c.	Data Mark	242	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.

d.	Break	243	NVT character BRK.
e.	Interrupt Process	244	The function IP.
f.	Abort output	245	The function AO.
g.	Are You There	246	The function AYT.
h.	Erase character	247	The function EC.
i.	Erase Line	248	The function EL.
j.	Go ahead	249	The GA signal.
k.	SB	250	Indicates that what follows is subnegotiation of the indicated option.
l.	WILL (option code)	251	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.
m.	WON'T (option code)	252	Indicates the refusal to perform, or continue performing, the indicated option.
n.	DO (option code)	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
o.	DON'T (option code)	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.
p.	IAC	255	Data Byte 255.

4.8 Connection establishment. The TELNET TCP connection is established between the user's port U and the server's port L. The server listens on its well known port L for such connections. Since a TCP connection is full duplex and identified by the pair of ports, the server can engage in many simultaneous connections involving its port L and different user ports U.

4.8.1 Port assignment. When used for remote user access to service hosts (i.e., remote terminal access) this protocol is assigned server port 23 (27 octal). That is L=23.

Custodians:
Army - CR
Navy - OM
Air Force - 90

Review Activities:
Army - SC, CR, AD
Navy - AS, YD, MC, OM, ND, NC, EC, SA
Air Force - 01, 02, 11, 13, 17, 99, 90

Preparing Activity:
DCA - DC

(Project IPSC-0176-01

Other Interest:
NSA - NS
JTCO - TT

APPENDIX A

TELNET BINARY TRANSMISSION

10. Command name and code.

TRANSMIT-BINARY 0

20. Command meanings.

20.1 IAC WILL TRANSMIT-BINARY. The sender of this command REQUESTS permission to begin transmitting, or confirms that it will now begin transmitting characters which are to be interpreted as 8 bits of binary data by the receiver of the data.

20.2 IAC WON'T TRANSMIT-BINARY. If the connection is already being operated in binary transmission mode, the sender of this command DEMANDS to begin transmitting data characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data. If the connection is not already being operated in binary transmission mode, the sender of this command REFUSES to begin transmitting characters which are to be interpreted as binary characters by the receiver of the data (i.e., the sender of the data demands to continue transmitting characters in its present mode). A connection is being operated in binary transmission mode only when one party has requested it and the other has acknowledged it.

20.3 IAC DO TRANSMIT-BINARY. The sender of this command REQUESTS that the sender of the data start transmitting, or confirms that the sender of data is expected to transmit, characters which are to be interpreted as 8 bits of binary data (i.e., by the party sending this command).

20.4 IAC DON'T TRANSMIT-BINARY. If the connection is already being operated in binary transmission mode, the sender of this command DEMANDS that the sender of the data start transmitting characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data (i.e., the party sending this command). If the connection is not already being operated in binary transmission mode, the sender of this command DEMANDS that the sender of data continue transmitting characters which are to be interpreted in the present mode. A connection is being operated in binary transmission mode only when one party has requested it and the other has acknowledged it.

30. Default.

WON'T TRANSMIT-BINARY

DON'T TRANSMIT-BINARY

The connection is not operated in binary mode.

40. Motivation for the option. It is sometimes useful to have available a binary transmission path within TELNET without having to utilize one of the more efficient, higher level protocols providing binary transmission (such as the File Transfer Protocol). The use of the IAC prefix within the basic TELNET protocol provides the option of binary transmission in a natural way, requiring only the addition of a mechanism by which the parties involved can agree to INTERPRET the characters transmitted over a TELNET connection as binary data.

50. Description of the option. With the binary transmission option in effect, the receiver should-interpret characters received from the transmitter which are not preceded with IAC as 8 bit binary data, with the exception of IAC followed by IAC which stands for the 8 bit binary data with the decimal value 255. IAC followed by an effective TELNET command (plus any additional characters required to complete the command) is still the command even with the binary transmission option in effect. IAC followed by a character which is not a defined TELNET command has the same meaning as IAC followed by NOP, although an IAC followed by an undefined command should not normally be sent in this mode.

60. Implementation suggestions. It is foreseen that implementations of the binary Transmission option will choose to refuse some other options (such as the EBCDIC transmission option) while the binary transmission option is in effect. However, if a pair of hosts can understand being in binary transmission mode simultaneous with being in, for example, echo mode, then it is all right if they negotiate that combination. The meanings of WON’T and DON’T are dependent upon whether the connection is presently being operated in binary mode or not. Consider a connection operating in EBCDIC mode which involves a system which has chosen not to implement any knowledge of the binary command. If this system were to receive a DO TRANSMIT-BINARY, it would not recognize the TRANSMIT-BINARY option and therefore would return a WON’T TRANSMIT-BINARY. If the default for the WON'T TRANSMIT-BINARY were always NVT ASCII, the sender of the DO TRANSMIT-BINARY would expect the recipient to have switched to NVT ASCII, whereas the receiver of the DO TRANSMIT-BINARY would not make this interpretation. Thus, we have the rule that when a connection is not presently operating in binary mode, the default (i.e., the interpretation of WON'T and DON'T) is to continue operating in the current mode, whether that is NVT ASCII, EBCDIC, or some other mode. This rule, however, is not applied once a connection is operating in a binary mode (as agreed to by both ends); this would require each end of the connection to maintain a stack, containing all of the encoding-method transitions which had previously occurred on the connection, in order to properly interpret a WON’T or DON’T. Thus, a WON’T or DON’T received after the connection is operating in binary mode causes the encoding method to revert to NVT ASCII.

70. Binary transmission mode. It should be remembered that a TELNET connection is a two way communication channel. The binary transmission mode must be negotiated separately for each direction of data flow, if that is desired. Implementation of the binary transmission option, as is the case with implementations of all other TELNET options, must follow the loop preventing rules given in the General Considerations section of the TELNET Protocol Specification. Consider now some issues of binary transmission both to and from both a process and a terminal:

70.1 Binary transmission from a terminal. The implementer of the binary transmission option should consider how (or whether) a terminal transmitting over a TELNET connection with binary transmission in effect is allowed to generate all eight bit characters, ignoring parity considerations, etc., on input from the terminal.

70.2 Binary transmission to a process. The implementer of the binary transmission option should consider how (or whether) all characters are passed to a process receiving over a connection with binary transmission in effect. As an example of the possible problem, TOPS-20 intercepts certain characters (e.g., ETX, the terminal control-C) at monitor level and does not pass them to the process.

70.3 Binary transmission from a process. The implementer of the binary transmission option should consider how (or whether) a process transmitting over a connection with binary transmission in effect is allowed to send all eight bit characters with no characters intercepted by the monitor and changed to other characters. An example of such a conversion may be found in the TOPS-20 system where certain non-printing characters are normally converted to a Circumflex (up-arrow) followed by a printing character.

70.4 Binary transmission to a terminal. The implementer of the binary transmission option should consider how (or whether) all characters received over a connection with binary transmission in effect are sent to a local terminal. At issue may be the addition of timing characters normally inserted locally, parity calculations, and any normal code conversion.

APPENDIX B

TELNET ECHO OPTION

10. Command name and code.

ECHO 1

20. Command meanings.

20.1 IAC WILL ECHO. The sender of this command REQUESTS to begin, or confirms that it will now begin, echoing data characters it receives over the TELNET connection back to the sender of the data characters.

20.2 IAC WON'T ECHO. The sender of this command DEMANDS to stop, or refuses to start, echoing the data characters it receives over the TELNET connection back to the sender of the data characters.

20.3 IAC DO ECHO. The sender of this command REQUESTS that the receiver of this command begin echoing, or confirms that the receiver of this command is expected to echo, data characters it receives over the TELNET connection back to the sender.

20.4 IAC DON'T ECHO. The sender of this command DEMANDS the receiver of this command stop, or not start, echoing data characters it receives over the TELNET connection.

30. Default.

WON'T ECHO

DON'T ECHO

No echoing is done over the TELNET connection.

40. Motivation for the option. The NVT has a printer and a keyboard which are nominally interconnected so that “echoes” need never traverse the network; that is to say, the NVT nominally operates in a mode where characters typed on the keyboard are (by some means) locally turned around and printed on the printer. In highly interactive situations it is appropriate for the remote process (command language interpreter, etc.) to which the characters are being sent to control the way they are echoed on the printer. In order to support such interactive situations, it is necessary that there be a TELNET option to allow the parties at the two ends of the TELNET connection to agree that characters typed on an NVT keyboard are to be echoed by the party at the other end of the TELNET connection.

50. Description of the option. When the echoing option is in effect, the party at the end performing the echoing is expected to transmit (echo) data characters it receives back to the sender of the data characters. The option does not require that the characters echoed be exactly the characters received (for example, a number of systems echo the ASCII ESC character with something other than the ESC character). When the echoing option is not in effect, the receiver of data characters should not echo them back to the sender; this, of course, does not prevent the receiver from responding to data characters received. The normal TELNET connection is two way. That is, data flows in each direction on the connection independently; and neither, either, or both directions may be operating simultaneously in echo mode. There are five reasonable modes of operation for echoing on a connection pair:

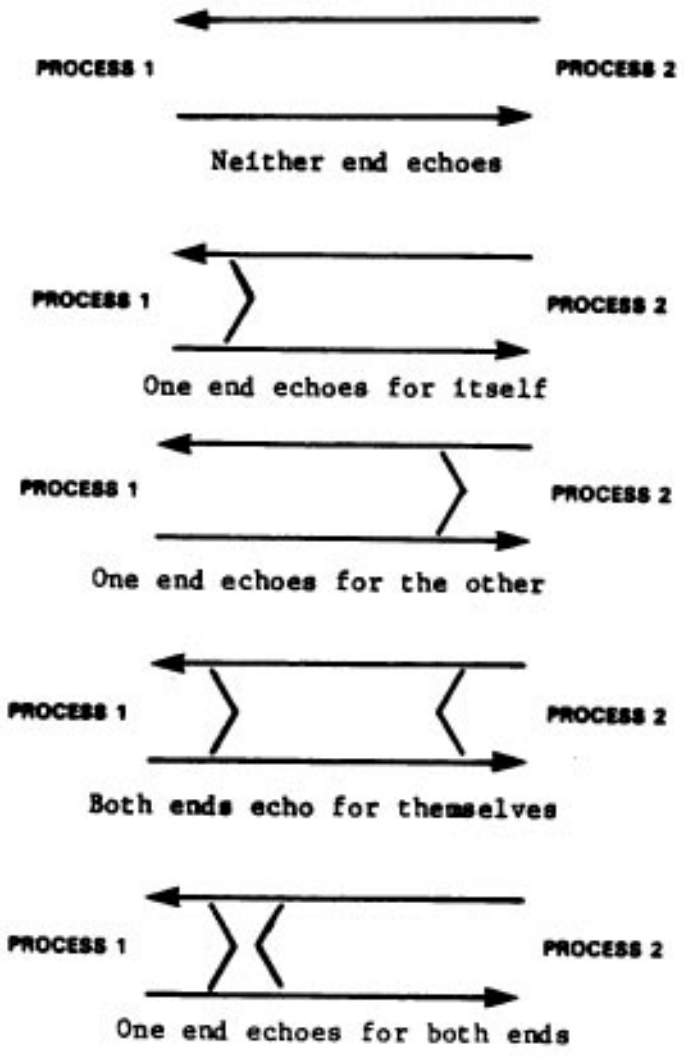

FIGURE 1. Five reasonable modes of operation for echoing on a connection pair.

This option provides the capability to decide on whether or not either end will echo for the other. It does not, however, provide any control over whether or not an end echoes for itself; this decision must be left to the sole discretion of the systems at each end (although they may use information regarding the state of "remote" echoing negotiations in making this decision). If BOTH hosts enter the mode of echoing characters transmitted by the other host, then any character transmitted in either direction will be "echoed" back and forth indefinitely. Therefore, care should be taken in each implementation that if one site is echoing, echoing is not permitted to be turned on at the other. As discussed in Section 4, both parties to a full-duplex TELNET connection initially assume each direction of the connection is being operated in the default mode which is non-echo (non-echo is not using this option, and the same as DON'T ECHO, WON'T ECHO). If either party desires himself to echo characters to the other party or for the other party to echo characters to him, that party gives the appropriate command (WILL ECHO or DO ECHO) and waits (and hopes) for acceptance of the option. If the request to operate the connection in echo mode is refused, then the connection continues to operate in non-echo mode. If the request to operate the connection in echo mode is accepted, the connection is operated in echo mode. After a connection has been changed to echo mode, either party may demand that it revert to non-echo mode by giving the appropriate DON'T ECHO or WON'T ECHO command (which the other party must confirm thereby allowing the connection to operate in non-echo mode). Just as each direction of the TELNET connection may be put in remote echoing mode independently, each direction of the TELNET connection must be removed from remote echoing mode separately.

60 Implementation of the option. Implementations of the echo option, as implementations of all other TELNET options, must follow the loop preventing rules given in the General Requirements section of the TELNET Protocol Standard. Also, so that switches between echo and non-echo mode can be made with minimal confusion (momentary double echoing, etc.), switches in mode of operation should be made at times precisely coordinated with the reception and transmission of echo requests and demands. For instance, if one party responds to a DO ECHO with a WILL ECHO, all data characters received after the DO ECHO should be echoed and the WILL ECHO should immediately precede the first of the echoed characters. The echoing option alone will normally not be sufficient to effect what is commonly understood to be remote computer echoing of characters typed on a terminal keyboard--the SUPPRESS-GO AHEAD option will normally have to be invoked in conjunction with the ECHO option to effect character-at-a-time remote echoing.

60.1 A sample implementation of the option. The following is a description of a possible implementation for a simple user system called "UHOST". A possible implementation could be that for each user terminal, the UHOST would keep three state bits: whether the terminal echoes for itself (UHOST ECHO always) or not (ECHO mode possible), whether the (human) user prefers to operate in ECHO mode or in non-ECHO mode, and whether the connection from this terminal to the server is in ECHO or non-ECHO mode. We will call these three bits P(hysical), D(esired), and A(ctual). When a terminal dials up the UHOST the P-bit is set appropriately, the D-bit is set equal to it, and the A-bit is set to non-ECHO. The P-bit and D-bit may be manually reset by

direct commands if the user so desires. For example, a user in Hawaii on a "full-duplex" terminal, would choose not to operate in ECHO mode, regardless of the preference of a mainland server. He should direct the UHOST to change his D-bit from ECHO to non-ECHO. When a connection is opened from the UHOST terminal to a server, the UHOST would send the server a DO ECHO command if the MIN (with non-ECHO less than ECHO) of the P- and D-bits is different from the A-bit. If a WON'T ECHO or WILL ECHO arrives from the server, the UHOST will set the A-bit to the MIN of the received request, the P-bit, and the D-bit. If this changes the state of the A-bit, the UHOST will send off the appropriate acknowledgment; if it does not, then the UHOST will send off the appropriate refusal if not changing meant that it had to deny the request (i.e., the MIN of the P-and D-bits was less than the received A-request). If while a connection is open, the UHOST terminal user changes either the P-bit or D-bit, the UHOST will repeat the above tests and send off a DO ECHO or DON'T ECHO, if necessary. When the connection is closed, the UHOST would reset the A-bit to indicate UHOST echoing. While the UHOST's implementation would not involve DO ECHO or DON'T ECHO commands being sent to the server except when the connection is opened or the user explicitly changes his echoing mode, bigger hosts might invoke such mode switches quite frequently. For instance, while a line-at-a-time system were running, the server might attempt to put the user in local echo mode by sending the WON'T ECHO command to the user; but while a character-at-a-time system were running, the server might attempt to invoke remote echoing for the user by sending the WILL ECHO command to the user. Furthermore, while the UHOST will never send a WILL ECHO command and will only send a WON'T ECHO to refuse a server sent DO ECHO command, a server host might often send the WILL and WON'T ECHO commands.

APPENDIX C

TELNET SUPPRESS GO AHEAD OPTION

10. Command name and code.

SUPPRESS-GO-AHEAD 3

20. Command meanings.

20.1 IAC WILL SUPPRESS-GO-AHEAD. The sender of this command requests permission to begin suppressing transmission of the TELNET GO AHEAD (GA) character when transmitting data characters, or the sender of this command confirms it will now begin suppressing transmission of GAs with transmitted data characters.

20.2 IAC WON'T SUPPRESS-GO-AHEAD. The sender of this command demands to begin transmitting, or to continue transmitting, the GA character when transmitting data characters.

20.3 IAC DO SUPPRESS-GO-AHEAD. The sender of this command requests that the sender of data start suppressing GA when transmitting data, or the sender of this command confirms that the sender of data is expected to suppress transmission of GAs.

20.4 IAC DON'T SUPPRESS-GO-AHEAD. The sender of this command demands that the receiver of the command start or continue transmitting GAs when transmitting data.

30. Default.

WON’T SUPPRESS-GO-AHEAD

DON'T SUPPRESS-GO-AHEAD

Go aheads are transmitted.

40. Motivation for the option. While the NVT nominally follows a half duplex protocol complete with a GO AHEAD signal, there is no reason why a full duplex connection between a full duplex terminal and a host optimized to handle full duplex terminals should be burdened with the GO AHEAD signal. Therefore, it is desirable to have a TELNET option with which parties involved can agree that one or the other or both should suppress transmission of GO AHEADS.

50. Description of the option. When the SUPPRESS-GO-AHEAD option is in effect on the connection between a sender of data and the receiver of the data, the sender need not transmit GAs. It seems probable that the parties to the TELNET connection will suppress GO AHEAD in both directions of the TELNET connection if GO AHEAD is suppressed at all; but, nonetheless, it

must be suppressed in both directions independently. With the SUPPRESS-GO-AHEAD option in effect, the IAC GA command should be treated as a NOP if received, although IAC GA should not normally be sent in this mode.

60. Implementation considerations. As the SUPPRESS-GO-AHEAD option is sort of the opposite of a line-at-a time mode, the sender of data which is suppressing GO AHEADs should attempt to actually transmit characters as soon as possible (i.e., with minimal buffering) consistent with any other agreements which are in effect. In many TELNET implementations it will be desirable to couple the SUPPRESS-GO-AHEAD option to the echo option so that when the echo option is in effect, the SUPPRESS-GO-AHEAD option is in effect simultaneously: both of these options will normally have to be in effect simultaneously to effect what is commonly understood to be character-at-a time echoing by the remote computer.

APPENDIX D

TELNET STATUS OPTION

10. Command name and code.

STATUS 5

20.1 Command meanings. This option applies separately to each direction of data flow.

20.1 IAC WILL STATUS. The sender of WILL status agrees to send status information, spontaneously or in response to future requests.

20.2 IAC WON'T STATUS. Sender refuses to carry on any further discussion of the current status of options.

20.3 IAC D0 STATUS. The sender of DO wishes to be able to send request for status of in option information or confirms that he is willing to send such requests.

20.4 IAC DON'T STATUS. Sender refuses to carry on any further discussion of the current status of options.

20.5 IAC SB STATUS SEND IAC SE. Sender requests receiver to transmit his (the receiver's) perception of the current status of TELNET options. The code for SEND is 1.

20.6 IAC SB STATUS IS... IAC SE. Sender is stating his perception of the current status of TELNET options. The code for IS is 0.

30. Default.

DON’T STATUS, WON’T STATUS

The current status of options will not be discussed.

40. Motivation for the option. This option allows a user/process to verify the current status of TELNET options (e.g., echoing) as viewed by the person/process on the other end of the TELNET connection. Simply renegotiating options could lead to the nonterminating request loop problem discussed in paragraph 4.2.3. This option fits into the normal structure of TELNET options by deferring the actual transfer of status information to the SB command.

50. Description of the option. WILL and DO are used only to obtain and grant permission for future discussion. The actual exchange of status information occurs within option subcommands (IAC SB STATUS...). Once the two hosts have exchanged a WILL and a DO, the sender of the WILL STATUS is free to transmit status information, spontaneously or in response to a request from

the sender of the DO. At worst, this may lead to transmitting the information twice. Only the sender of the DO may send requests (IAC SB STATUS SEND IAC SE) and only the sender of the WILL may transmit actual status information (within an IAC SB STATUS IS ... IAC SE command). IS has the subcommands WILL, DO and SB. They are used EXACTLY as used during the actual negotiation of TELNET options, except that SB is terminated with SE, rather than IAC SE. Transmission of SE, as a regular data byte, is accomplished by doubling the byte (SE SE). Options that are not explicitly described are assumed to be in their default states. A single IAC SB STATUS IS... IAC SE describes the condition of ALL options.

60. Example of option application. The following is an example of use of the option:

Hostl: IAC DO STATUS

Host2: IAC WILL STATUS

(Host2 is now free to send status information at any time Solicitations from Hostl are NOT necessary. This should not produce any dangerous race conditions. At worst, two IS's will be sent.)

Hostl (perhaps): IAC SB STATUS SEND IAC SE

Host2 (the following stream is broken-into multiple lines only for readability. No carriage returns are implied.):

IAC SB STATUS IS
WILL ECHO
00 SUPPRESS-GO-AHEAD
WILL STATUS
DO STATUS
IAC SE

Explanation of Host2’s perceptions: It is responsible for echoing back the data characters it receives over the TELNET connection; it will not send Go-Ahead signals; it will both issue and request Status information.

APPENDIX E

TELNET TIMTNG MARK OPTION

10. Command name and code.

TIMING-MARK 6

20. Command meanings.

20.1 IAC WILL TIMING-MARK. The sender of this command ASSURES the receiver of this command that it is inserted in the data stream at the “appropriate place” to insure synchronization with a DO TIMING-MARK transmitted by the receiver of this command.

20.2 IAC WON'T TIMING-MARK. The sender of this command REFUSES to insure that this command is inserted in the data stream at the "appropriate place" to insure synchronization.

20.3 IAC D0 TIMING-MARK. The sender of this command REQUESTS that the receiver of this command return a WILL TIMING-MARK in the data stream at the "appropriate place" as defined in paragraph 11.4 below.

20.4 IAC DON'T TIMING-MARK. The sender of this command notifies the receiver of this command that a WILL TIMING-MARK (previously transmitted by the receiver of this command) has been IGNORED.

30. Default.

WON’T TIMING-MARK, DON’T TIMING-MARK

i.e., No explicit attempt is made to synchronize the activities at the two ends of the TELNET connection.

40. Motivation for the option. It is sometimes useful for a user or process at one end of a TELNET connection to be sure that previously transmitted data has been completely processed, printed, discarded, or otherwise disposed of. This option provides a mechanism for doing this. In addition, even if the option request (DO TIMING-MARK) is refused (by WON'T TIMING-MARK) the requester is at least assured that the refuser has received (if not processed) all previous data.

50. Examples of option application. As an example of a particular application, imagine a TELNET connection between a physically full duplex terminal and a "full duplex" server system which permits the user to “type ahead” while the server is processing previous user input. Suppose that both sides have agreed to Suppress Go Ahead and that the server has agreed to provide echoes. The server now discovers a command which it cannot parse, perhaps because of a user typing error. It would like to throw away all of the user’s “type-ahead” (since failure of the parsing of one command is likely

to lead to incorrect results if subsequent commands are executed), send the user an error message, and resume interpretation of commands which the user typed after-seeing the error message. If the user were local, the system would be able to discard the buffered input; but input may be buffered in the user's host or elsewhere. Therefore, the server might send a DO TIMING-MARK and hope to receive a WILL TIMING-MARK from the user at the “appropriate place” in the data stream. The “appropriate place” (in absence of other information) is clearly just before the first character which the user typed after seeing the error message. That is, it should appear that the timing mark was “printed” on the user’s terminal and that, in response, the user typed an answering timing mark. Next, suppose hat the user in the example above realized that he had misspelled a command, realized that the server would send a DO TIMING-MARK, and wanted to start “typing ahead” again without waiting for this to occur. He might then instruct his own system to send a WILL TIMING-MARK to the server and then begin “typing ahead” again. (Implementers should remember that the user's own system must remember that it sent the WILL TIMING-MARK so as to discard the DO/DON'T TIMING-MARK when it eventually arrives.) Thus, in this case the “appropriate place” for the insertion of the WILL TIMING-MARK is the place defined by the user. In both of the examples above, it is the responsibility of the system which transmits the DO TIMING-MARK to discard any unwanted characters; the WILL TIMING-MARK only provides help in deciding which characters are “unwanted”.

60. Description of the option. Suppose that Process A of Figure 2 wishes to synchronize with B. The DO TIMING-MARK is sent from A to B. B can refuse by replying WON’T TIMING-MARK, or agree by permitting the timing mark to flow through his “outgoing” buffer, BUF2. Then, instead of delivering it to the terminal, B will enter the mark into his “incoming” buffer BUF1, to flow through toward A. When the mark has propagated through B’s incoming buffer, B returns the WILL TIMING-MARK over the TELNET connection to A.

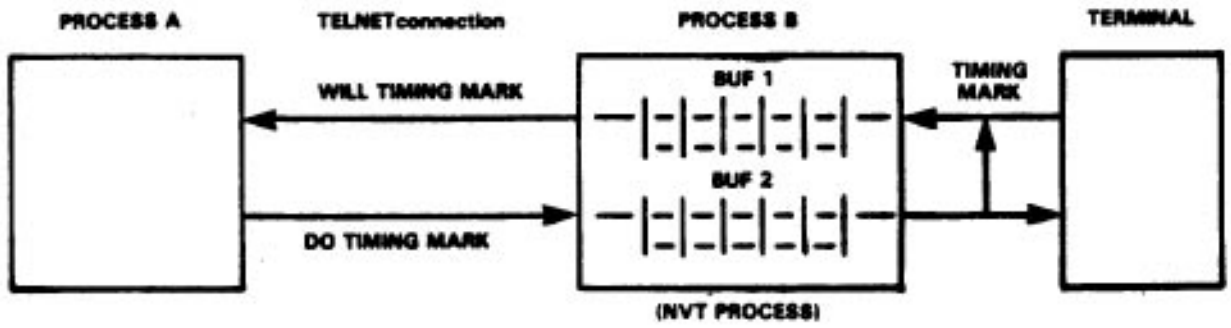

FIGURE 2. Synchronization of processes.

When A receives the WILL TIMING-MARK, he knows that all the information he sent to B before sending the timing mark been delivered, and all the information sent from B to A before turnaround of the timing mark has been delivered.

70. Three typical applications.

70.1 Round-trip delay. Measure round-trip delay between a process and a terminal or another process.

70.2 Resynchronization. Resynchronizing an interaction is described in paragraph 4 above. A is a process interpreting commands forwarded from a terminal by B. When A sees an illegal command it:

a. Sends <carriage return>, <line feed>, <question mark>.

b. Sends DO TIMING-MARK.

c. Sends an error message.

d. Starts reading input and throwing it away until it receives a WILL TIMING-MARK.

e. Resumes interpretation of input.

This achieves the effect of flushing all “type ahead” after the erroneous command, up to the point when the user actually saw the question mark.

70.3 Dual synchronization. The dual of paragraph 70.2. The terminal user wants to throw away unwanted output from A.

a. B sends DO TIMING-MARK, followed by some new command.

b. B starts reading output from A and throwing it away until it receives WILL TIMING-MARK.

c. B resumes forwarding A’s output to the terminal.

This achieves the effect of flushing all output from A, up to the point where A saw the timing mark, but not output generated in response to the following command.

APPENDIX F

TELNET EXTENDED OPTIONS - LIST OPTION

10. Command name and code.

EXTENDED-OPTIONS-LIST (EXOPL) 255

20. Command meanings.

20.1 IAC WILL EXOPL. The sender of this command REQUESTS permission to begin negotiating, or confirms that it will begin negotiating, TELNET options which are on the “Extended Options List.”

20.2 IAC WON'T EXOPL. The sender of this command REFUSES to negotiate, or to continue negotiating, options on the “Extended Options List.”

20.3 IAC DO EXOPL. The sender of this command REQUESTS that the receiver of this command begin negotiating, or confirms that the receiver of this command is expected to begin negotiating, TELNET options which are on the “Extended Options List.”

20.4 IAC DON'T EXOPL. The sender of this command DEMANDS that the receiver conduct no further negotiation of options on the “Extended Options List.”

20.5 IAC SB EXOPL <subcommand>. The subcommand contains information required for the negotiation of an option of the “Extended Options List.” The format of the subcommand is discussed in paragraph 5 below.

30. Default.

WON’T EXOPL, DON’T EXOPL
Negotiation of options on the “Extended Options List” is not permitted.

40. Motivation for the option. Eventually, a 257th TELNET option will be needed. This option will extend the option list for another 256 options in a manner which is easy to implement. The option is proposed now, rather than later (probably much later), in order to reserve the option number (255).

50. Description of the option. The EXOPL option has five subcommand codes: WILL, WON’T, DO, DON’T, and SB. They have exactly the same meanings as the TELNET commands with the same names, and are used in exactly the same way. For consistency, these subcommand codes will have the same values as the TELNET command codes (250-254). Thus, the format for negotiating a specific option on the “Extended Options List” (once both parties have agreed to use it) is:

IAC SB EXOPL DO/DON’T/WILL/WON’T/<option code> IAC SE

Once both sides have agreed to use the specific option specified by <option code>, subnegotiation may be required. In this case the format to be used is:

IAC SB EXOPL SB <option code> <parameters> SE IAC SE

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

(See Instructions – Reverse Side)

1. DOCUMENT NUMBER	2. DOCUMENT TITLE
MIL-STD-1782	

3a. NAME OF SUBMITTING ORGANIZATION	4. TYPE OF ORGANIZATION *(Mark one)*
	☐ VENDOR ☐ USER ☐ MANUFACTURER ☐ OTHER *(Specify)*: _____
b. ADDRESS *(Street, City, State, ZIP Code)*	

5. PROBLEM AREAS

a. Paragraph Number and Wording:

b. Recommended Wording:

c. Reason/Rationale for Recommendation:

6. REMARKS

7a. NAME OF SUBMITTER *(Last, First, MI)* – Optional	b. WORK TELEPHONE NUMBER *(Include Area Code)* – Optional
c. MAILING ADDRESS *(Street, City, State, ZIP Code)* – Optional	8. DATE OF SUBMISSION *(YYMMDD)*

DD FORM 82 MA 1426

PREVIOUS EDITION IS OBSOLETE.